เขียนไทย

บุญเลิศ เอี่ยมทัศนา

ปาสคาล อรรถกร

ชื่อเรื่องในชุดนี้ของผมหากจะเขียนอย่างยาวคงต้องเขียนว่า **โปรแกรมอรรถประโยชน์ในภาษาปาสคาล** โดยผมมีแนวความคิดที่จะนำโปรแกรมย่อยที่เป็นโปรแกรมอรรถประโยชน์(ultility) มาเสนอต่อคุณผู้อ่านเพื่อเก็บไว้เป็นเครื่องมือในการเขียนโปรแกรมใช้งานจริงๆ ของคุณต่อไป ดังที่ฝรั่งเขารวมกันไว้เป็นชุดให้ชื่อว่า tool box ต่างๆ สำหรับของผมจะทำในเรื่องที่เกี่ยวกับภาษาไทยและในเรื่องที่ฝรั่งไม่สนใจ แต่เมื่อเขียนไปก็คงจะพาดพิงไปถึงเรื่องอื่นๆ ด้วย

บทความของผมในชุดนี้จะเป็นบทความที่จบสมบูรณ์เป็นตอนๆ ไป นำไปใช้งานได้เลย แต่คงจะไม่มีในทุกฉบับ สุดแต่ว่าจะนึกเรื่องใดออก และที่ว่าเป็นปาสคาลนั้น ผมหมายถึงเทอร์โบปาสคาลรุ่น 3.0 และโดยเฉพาะบนเครื่องไอบีเอ็มพีซี สำหรับคุณที่ใช้ปาสคาลแบบอื่นบนเครื่องอื่นนั้น คงจะต้องนำไปดัดแปลงด้วยตนเองครับ

ตอนแรกที่ผมจะนำเสนอนี้เป็นการทำภาษาไทย หรือจะกล่าวให้ชัดเจนคือ จะเป็นการสร้างโปรแกรมย่อย เพื่อนำไปใช้ในโปรแกรมใช้งานจริง เพื่อให้โปรแกรมนั้นสามารถอ่านภาษาไทยจากคีย์บอร์ด และพิมพ์ภาษาไทยออกทางจอภาพได้ (สำหรับเครื่องพิมพ์จะกล่าวในภายหลัง) หากคุณสงสัยว่ามีการทำภาษาไทยบนเครื่องแล้ว ทำไมจึงต้องการโปรแกรมอรรถประโยชน์ในเรื่องนี้อีก ก็ขอเรียนว่าเป็นคนละวิธีการกัน และผมคิดว่าวิธีนี้เป็นวิธีที่เหมาะที่สุดสำหรับภาษาปาสคาล และเมื่อคอมพายล์เป็นไฟล์ .COM แล้ว ผู้ใช้จะรู้สึกเหมือนใช้โปรแกรมที่เป็นภาษาไทยโดยธรรมชาติ ซึ่งให้ความรู้สึกที่ดีกว่า

โดยหลักการ

โดยหลักการแล้ว การทำภาษาไทยให้กับเครื่องไมโครคอมพิวเตอร์ไม่แตกต่างกันนัก ปัญหามีอยู่แต่เพียงว่า ทำอย่างไรถึงจะอ่านคีย์บอร์ดเป็นอักษรไทยได้ ทำอย่างไรถึงจะแสดงอักษรไทยบนจอภาพได้ และทำอย่างไรถึงจะนำออกเครื่องพิมพ์ได้ แต่ปัญหาที่หนักที่สุดคือ จะแทรกโปรแกรมที่เราเขียนขึ้น เพื่อดำเนินการในเรื่องนี้เข้าไปในของเดิมได้อย่างไร

ก็ต้องดูกันที่ระดับของการทำภาษาไทย อาจจะแบ่งได้หลายระดับ แต่โดยหลักใหญ่มีอยู่เพียง 2 ระดับคือ ทำในระบบจัดการกับทำในโปรแกรมใช้งาน การทำในระบบจัดการเป็นเรื่องที่ยากพอสมควร ซึ่งอาจารย์ยืน ภู่วรวรรณ ได้นำมากล่าวถึงและคงจะกล่าวถึงในรายละเอียดต่อไป

เขียนไทย

สำหรับการทำในโปรแกรมใช้งาน ซึ่งผมขอหมายรวมถึงโปรแกรมตัวแปรภาษาเช่น ปาสคาลด้วย ความยากง่ายขึ้นอยู่กับโปรแกรมนั้นๆ สำหรับเทอร์โบปาสคาลโดยเฉพาะรุ่น 3.0 แล้วนับว่าง่ายมาก แต่ผมขอทำความเข้าใจก่อนว่า การทำภาษาไทยนี้เป็นเพียงการสร้างโปรแกรมย่อยดังกล่าว ไม่ได้หมายความว่าโปรแกรมนี้ จะเป็นโปรแกรมประเภทเวิร์ดสตาร์ ซึ่งหากคุณต้องการจะต้องดำเนินการต่อไปเอง และคงจะพอมองเห็นแนวทางบ้างจากบทความตอนนี้

คีย์บอร์ด

ผมยกคีย์บอร์ดเป็นปัญหาแรกก็จะเริ่มจากคีย์บอร์ด ผมคิดว่าคงไม่มีใครคิดจะซื้อคีย์บอร์ดมาอีก 1 ชุด เพื่อใช้เป็นคีย์ภาษาไทย เราคงใช้คีย์ภาษาอังกฤษเดิม แต่ทำให้มันสามารถส่งรหัสภาษาไทยแก่เราได้เมื่อเราต้องการ เพราะความจริงการทำภาษาไทยคือการทำภาษาไทย/อังกฤษ คือมีทั้ง 2 ภาษา

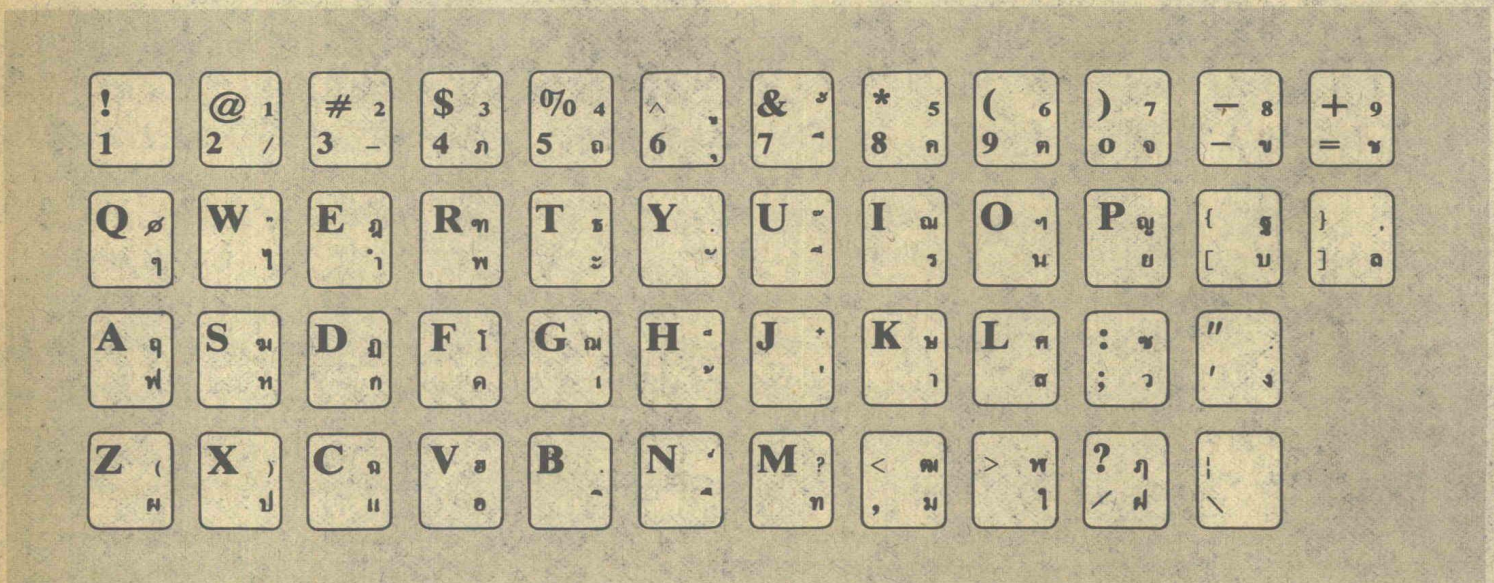
ขอให้ดูในรูปที่ 1 เป็นรูปแบบของการวางแป้นอักษรทั้งไทยและอังกฤษบนคีย์เดียวกัน และรูปที่ 2 ซึ่งเป็นตารางรหัสอักขระทั้งไทยและอังกฤษ ผมขอพูดอย่างรวบรัดว่าเมื่อเราใช้คำสั่ง Read (KBD,ch) ในปาสคาล และผู้ใช้กดคีย์ A ใหญ่รหัสอักขระที่ตัวแปร ch ได้รับคือ $41 (เป็นเลขฐานสิบหก) หากผู้ใช้กดคีย์ a เล็ก ch จะได้รับค่า $61

หน้าที่ของเราคือ ทำอย่างไรเมื่อผู้ใช้กดคีย์ A ใหญ่ แต่จะให้ ch ได้รับค่า $C2 ซึ่งหมายถึง ฤ ถ้ากด a เล็กให้ ch ได้ค่า $BD ซึ่งหมายถึง ฟ และคีย์อื่นๆ ด้วยครับขอให้ดูรูปที่ 1 และ 2 ประกอบ เรื่องนี้ไม่ยาก เราเพียงแต่ทำตารางรหัสอักขระภาษาอังกฤษตั้งแต่ $20 ถึง $7E และเมื่อคีย์ภาษาอังกฤษตัวใดตรงกับคีย์ภาษาไทยตัวใด เราก็ใส่รหัสภาษาไทยให้ตรงกัน ผมไม่มีรูปให้ดูเพราะผมไม่ได้ทำเอง ขอมาจากคุณนิพันธ์ วงศ์-ทองศิริ แต่ตารางนี้จะไปปรากฏในโปรแกรมซึ่งผมจะกล่าวถึงต่อไป

ถึงตรงนี้ผมขอทำความเข้าใจไว้ขั้นหนึ่งก่อนว่า ทุกครั้งที่อ่านคีย์ ค่าที่ได้มาจะเป็นรหัสอักขระภาษาอังกฤษเสมอ จึงเป็นหน้าที่ของเราที่จะเขียนโปรแกรม แปลงค่าคีย์จากอังกฤษเป็นไทย ด้วยการเปิดตารางที่เตรียมไว้ แต่เราจะทราบได้อย่างไรว่าผู้ใช้ต้องการให้เป็นอังกฤษหรือไทย

จะทำได้โดยหาคีย์ที่จะไม่ใช้ในโปร-แกรมของคุณมาสัก 2 คีย์ สมมติว่าเป็นคีย์ TT กับ EE แล้วบอกผู้ใช้ว่า ถ้าต้องการอักษรไทยให้กดคีย์ TT หากต่อไปต้องการเปลี่ยนกลับเป็นภาษาอังกฤษให้กดคีย์ EE แต่เจ้า 2 คีย์นี้หายากจริงๆ ครับ ส่วนมากจึงกำหนดไว้เพียงคีย์เดียว โดยกำหนดให้ว่ากดครั้งหนึ่งเป็นไทย กดอีกครั้งเป็นอังกฤษสลับกันไปมา(toggle)

คีย์ที่นิยมใช้กันพอสมควรคือคีย์ ~ แต่ในโครงงานนี้ผมจะใช้คีย์ CTRL-T คือกดคีย์ CTRL แล้วกดคีย์ T จะได้รหัสอักขระเป็น $14 ซึ่งผมไม่ได้กล่าวแต่แรกว่ารหัสอักขระตั้งแต่ $00 ถึง $1F รวม 32 รหัส เขาจัดเป็นพวกคีย์ควบคุมการทำงานของจอภาพ เครื่องพิมพ์และอื่นๆ รายละเอียดขอให้ดูในตารางรหัส ASCII(ท้ายเล่มของหนังสือ "เรียนรู้ภาษาปาสคาล" ก็มีครับ) อักขระนอกนี้โดย



รูปที่ 1 แสดงการกำหนดคีย์บอร์ดไทย/อังกฤษ การวางคีย์ใกล้เคียงกับความเป็นจริง ขอให้เทียบกับคีย์บอร์ดตัวจริงด้วย

เขียนไทย

ปกติไม่นำออกแสดงบนจอภาพ

จอภาพ

เป็นอันว่าเราได้รหัสอักขระภาษาไทยอยู่ในตัวแปร ch แล้ว แต่ยังไม่มีภาพปรากฏบนจอภาพ (เพราะเป็นความประสงค์ที่จะยังไม่ให้แสดงภาพอักขระจึงใช้คำสั่ง Read (KBD, ch)) งานขั้นต่อไปของเราคือนำออกแสดงบนจอภาพ แต่จะให้แสดงในแบบใดคือในโหมดคาแรกเตอร์หรือกราฟิก ตามความเห็นของผมถ้าในการใช้งานตามปกติเดิม เขาแสดงภาพอักขระภาษาอังกฤษในโหมดใด เราก็ควรแสดงภาพอักขระภาษาไทยในโหมดนั้นด้วย

สำหรับเครื่องไอบีเอ็มพีซี แสดงในโหมดคาแรกเตอร์ ก็ต้องเดือดร้อนถึงคุณที่ต้องนำบอร์ดโมโนโครมไปให้ร้านจำหน่ายเขาติดตั้ง ROM ภาษาไทยให้ เมื่อพูดถึง ROM ตัวนี้ก็ขอทำความเข้าใจเพิ่มคือ รหัสอักขระทั้งหมดมีอยู่ด้วยกัน 256 อักขระ คือตั้งแต่ $00 ถึง $FF แต่ใช้ในภาษาอังกฤษจริงๆ รวมทั้งพวกรหัสควบคุมด้วยเพียง 128 อักขระ คือตั้งแต่ $00 ถึง $7F ซึ่งผมขอเรียกว่าครึ่งล่าง สำหรับครึ่งบนคือตั้งแต่รหัส $80 ถึง $FF ไอบีเอ็มให้เป็นรหัสของตัวหนังสือกรีกบ้าง เครื่องหมายทางแคลคูลัสบ้าง และทำเป็นกรอบต่างๆ บ้าง ทดลองดูได้โดยใช้เครื่องหมาย # นำหน้ารหัสแล้วใช้คำสั่งเช่น Write(#$NN) โดย NN คือค่าตั้งแต่ $80 ถึง $FF จะปรากฏภาพแปลกๆ บนจอภาพ

ในการทำภาษาไทย เราใช้ครึ่งบนแสดงอักษรไทยดังรูปที่ 2 ใน ROM ภาษาไทยจึงมีทั้งภาษาไทยและภาษาอังกฤษครับ หากคุณตัดใจว่าไม่ต้องการเครื่องหมายและกรอบแปลกๆ นั้น ก็ถอด ROM ตัวเดิมแล้วใส่ ROM ภาษาไทยเข้าไปแทนที่ได้โดยตรง แต่ส่วนมากมักจะรักพี่เสียดายน้อง ทางร้านเขาจึงจับมันวางซ้อนกัน หรือไม่ก็เปลี่ยนเป็น ROM เบอร์ 2764 ทั้ง 2 แบบนี้จะมีสวิทช์ให้เราเลือกว่า จะใช้แบบเดิมหรือแบบที่มีภาษาไทย

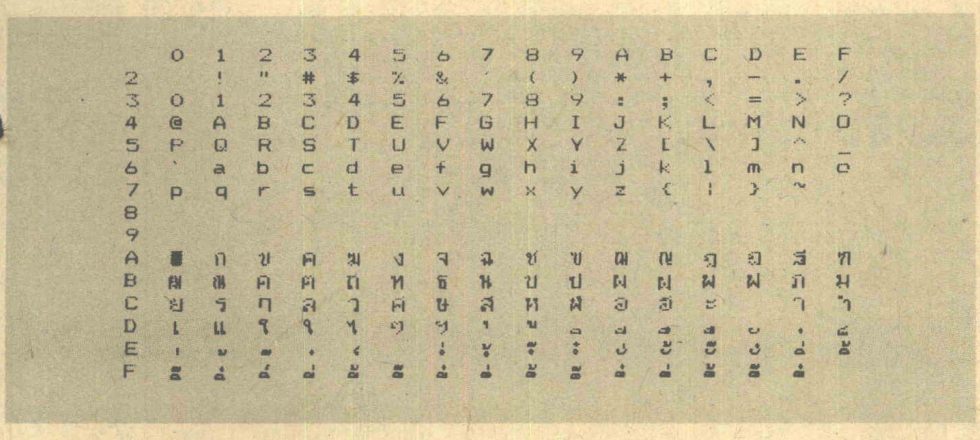
ถึงตอนนี้เราก็สามารถนำภาพอักษรไทยออกแสดงบนจอภาพได้แล้ว โดยใช้คำสั่ง Write แต่สระ วรรณยุกต์ และไม้ทัณฑฆาตจะเรียงอยู่บรรทัดเดียวกันกับพยัญชนะ ซึ่งบางท่านบอกว่าเป็นภาษาขอม ความจริงกลับตรงข้าม ภาษาไทยที่เราเขียนในปัจจุบันต่างหากที่เป็นขอม ภาษาไทยที่สระและพยัญชนะอยู่บรรทัดเดียวกัน คือภาษาไทยแท้ที่พ่อขุนรามคำแหงทรงบัญญัติ แต่เพราะความเคารพขอมอย่างสุดซึ้งในคนรุ่นหลัง ทำให้คนรุ่นเราต้องจำใจเขียนโปรแกรมจัดระดับอักษรไทยเป็น 3 หรือ 4 ระดับ

โปรแกรม

ในการเขียนโปรแกรม เราต้องตัดสินใจอีกว่าจะให้ออกจอภาพในแบบ 3 หรือ 4 ระดับ เช่นคำว่า มู่ลี่ หากเป็น 4 ระดับ จะได้สระอูระดับหนึ่ง ม และ ล ระดับหนึ่ง สระอีระดับหนึ่ง และไม้เอกอีกระดับหนึ่ง ที่นิยมกันคือ ให้ออกในแบบ 3 ระดับ โดยนำสระบนและวรรณยุกต์มารวมกัน และให้รหัสอักขระใหม่แก่อักขระใหม่นี้ คือรหัสตั้งแต่ $E6 ถึง $FE ในรูปที่ 2 ที่เราคุ้นกันในเครื่องพิมพ์ดีดคือ ะ ซึ่งผมจะใช้วิธีนี้

ดังที่กล่าวแล้วว่าเรามีปัญหาเกี่ยวกับคีย์บอร์ดกับจอภาพ ส่วนเรื่องของเครื่องพิมพ์ผมขอพักไว้ก่อน เราจึงต้องเขียนโปรแกรมย่อยเพื่อดำเนินการกับคีย์บอร์ด ซึ่งผมจะให้ชื่อว่า getc และกับจอภาพจะให้ชื่อว่า putc ทั้งสองโปรแกรมนี้ จะอ่านและเขียนภาษาไทย/อังกฤษ ครั้งละ 1 อักขระ แต่ในการใช้งานจริง เรามักจะอ่าน–เขียนข้อความเป็นสตริง จึงเพิ่มโปรแกรมย่อย ReadString กับ WrilString อีก 2 โปรแกรม

โปรแกรมย่อยทั้งหมดนี้ จะรวมไว้ในไฟล์ THULT.SBR เมื่อจะใช้งานจึงรวม (include) เข้ากับโปรแกรมของเรา ขอให้ดูตัวอย่างการใช้งานในโปรแกรม TestThai



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A	■	ก	ข	ค	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ญ	ฎ	ฏ	ฐ	ฑ
B	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ	ภ	ม
C	ย	ร	ฤ	ล	ว	ศ	ษ	ส	ห	ฬ	อ	ฮ	ะ		า	ำ
D	เ	แ	ใ	ไ	ฯ	ๆ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ
E	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ
F	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ	ฯ

◀ **รูปที่ 2 ตารางรหัสอักขระอังกฤษและไทย การอ่านรหัสให้อ่านจากซ้ายไปขวา เช่น A มีค่าเป็น $41 และ ก มีค่าเป็น $A1 ขอให้สังเกตด้วยว่าอักขระผสมสระและวรรณยุกต์เริ่มจาก $E6, $EA, $EE, $F3, $F7 และ $FB ซึ่งนำไปทำเป็นตาราง UPtab**

เขียนไทย

รูปที่ 3 จะเห็นว่าได้ใช้คอมพายเลอร์ไดเรกตีฟในกำรรวม(include) ไฟล์ THULT.SBR เข้ากับโปรแกรมนี้ สำหรับตัวโปรแกรม TestThai ก็เพียงแต่อ่านคีย์ที่ผู้ใช้ป้อนมาเก็บในตัวแปรสตริง(astr) แล้วนำออกแสดงทางจอภาพ ด้วยการเรียกใช้โปรแกรมย่อย ReadStr และ WriteStr ในไฟล์ THULT.SBR

อีกเรื่องหนึ่งคือ ได้ใช้คอมพายเลอร์ไดเรกตีฟ V ให้ผ่อนผันในการที่จะถ่ายค่าระหว่างสตริง ไม่เช่นนั้น astr ในเมนกับ stn ในโปรแกรมย่อย จะต้องมีขนาดเดียวกันจึงจะถ่ายค่าให้กันและกันได้ แต่การผ่อนผันนี้ก็มีข้อเสีย คือถ้าเรานำค่าสตริงที่ยาวกว่าหมายถึงมีจำนวนอักขระยาวกว่า ไปให้แก่ตัวแปรที่กำหนดขนาดของสตริงไว้สั้นกว่า จะเกิดการล้นสตริงข้ามไปให้ค่าแก่ตัวแปรตัวอื่นที่อยู่ถัดไป จึงเป็นเรื่องที่ต้องระวัง

```
B>type tst_tult.pas
PROGRAM TestThai(input,output);
TYPE   anyStr = String[50];

{$I thult.sbr}


{$V-}

var      astr : String[30];
begin  { MAIN }
  WriteLn('enter'); WriteLn;
  repeat
    ReadStr(astr); WriteLn; WriteLn;
    WriteStr(astr); WriteLn; WriteLn;
  until astr = '';
end.
```

รูปที่ 3 โปรแกรม TestThai แสดงการใช้คำที่สร้างใหม่ในไฟล์ THULT.SBR

getc

ดังที่กล่าวแล้วว่าโปรแกรมย่อยในชุด THULT.SBR มีอยู่ด้วยกัน 4 ตัว ตัวแรกที่จะกล่าวถึงคือ getc ดังแสดงในรูป A แต่ก่อนจะถึงตัวโปรแกรม ได้มีการกำหนดพวกตัวแปร และค่าคงตัวต่างๆ ไว้เป็นส่วนกลางของโปรแกรมย่อยในชุดนี้ (เมื่อนำไปรวมกับตัวโปรแกรมหลัก จะกลายเป็นส่วนกลางของโปรแกรมหลักด้วย ซึ่งต้องคอยระวัง) ที่ผมไม่กำหนดเป็นส่วนตัว(local) ของแต่ละโปรแกรม เพราะต้องการเขียนโปรแกรมให้สั้นเพื่อให้อ่านง่าย

ได้กำหนดขนาดของสตริงที่จะใช้ไว้เพียง 50 อักขระ ซึ่งคิดว่าน่าจะเป็นการเพียงพอ (หากคิดว่าไม่พอให้เพิ่มขนาดได้) ต่อไปได้กำหนดค่าคงตัวให้แก่คำต่างๆ เพื่อนำไปใช้ในโปรแกรม ผมขอกล่าวในตอนนี้

A

```
B>type thult.sbr
CONST  UNDR = #$D7; UPR = #$D9; BEEP = #$07;
       SPCE = #$20; BS  = #$08; RET  = #$0D; TOGGLE = ^T;
VAR    x,y,Upcntr : Byte;
       OldCh : Char;
       Tflg, GoodCh : Boolean;


PROCEDURE  getc(VAR ch : Char);
CONST  THtab : ARRAY[#$20..#$7F] OF Byte =
        ($20,$2E,$5F,$32,$33,$34,$EB,$A5,$36,$37,$35,$39,$BF,$A2,$D3,$BB,
         $A6,$C3,$2D,$2F,$BE,$B4,$D7,$DB,$A3,$B3,$A9,$C4,$B0,$A8,$C9,$DE,
         $31,$C2,$3F,$A7,$AC,$AD,$D2,$AA,$DF,$B1,$E3,$C6,$C5,$2C,$E4,$D6,
         $AB,$30,$AF,$A4,$B6,$E2,$CB,$22,$29,$E3,$28,$B8,$00,$C3,$D8,$38,
         $00,$BD,$D9,$D1,$A1,$CF,$B2,$DO,$E1,$C1,$EO,$CE,$C7,$B5,$DC,$B7,
         $CO,$D5,$BC,$C8,$CC,$DA,$CA,$D4,$B9,$DD,$BA,$AE,$00,$2C,$00,$00);

BEGIN
  Read(KBD,ch);
  IF ch = TOGGLE THEN
    BEGIN
    Tflg := NOT Tflg;
    Write(BEEP); ch := #$00;
    END;
  IF ( ch > SPCE ) AND  Tflg THEN
    ch := Chr(THtab[ch]);
END;
```

เลยว่าตัวเลขเช่น $4A ในเลขฐานสิบหกหรือคือ 74 ในเลขฐานสิบ ถ้าเขียนเช่นนี้หมายถึงเลขจำนวน เช่น ถ้าเราสั่งว่า Write($4A) หรือ Write(74) จะพิมพ์ตัวเลข 74 ออกมา แต่ถ้าเราเติมเครื่องหมาย # ไว้ข้างหน้าจะหมายถึงรหัสอักขระ เช่น สั่งว่า Write(#$4A) หรือ Write(#74) จะพิมพ์ภาพอักษร J ออกมา ในโปรแกรมนี้ผมจะใช้เลขฐานสิบหกเป็นหลัก เพราะง่ายต่อการทำความเข้าใจ

สำหรับโปรแกรม getc ซึ่งจะอ่านอักขระเพียง 1 อักขระ และไม่นำออกแสดงบนจอภาพ เริ่มด้วยการกำหนดตาราง THtab ซึ่งจะเป็นตารางแปลงค่ารหัสอักขระอังกฤษเป็นไทย โดยใช้อักขระในภาษาอังกฤษนั้นเองเป็นอินเด็กซ์จาก # $20 ถึง # $7F (ขอให้ดูรูปที่ 2 ประกอบด้วย) ผู้ทำตาราง THtab จะดูจากรูปที่ 1 ว่าอักขระภาษาอังกฤษใดตรงกับอักขระภาษาไทยใด นำไปเปิดตารางในรูปที่ 2 และใส่ค่าไล่ไปจากต้นตาราง

เริ่มต้นจาก # $20 เป็น space ในโหมดอังกฤษตรงกับวรรค ในภาษาไทยจึงใส่รหัสเดิมคือ # $20 แต่ในการใส่ค่าในตาราง ได้ใส่เป็นเลขจำนวนคือ $20 เพื่อไม่ต้องพิมพ์เครื่องหมาย # ถึง 96 ครั้ง จะเปลี่ยนเป็นรหัสอักขระในโปรแกรม ตัวต่อไปคือ # $21 เป็นเครื่องหมาย ! ในโหมดอังกฤษ แต่ในโหมดภาษาไทยนิยมให้เป็นจุด จึงใส่ค่า $2E ขอให้ดูในบรรทัดที่สอง ตัวแรกจะเป็น # $30 คือเลขศูนย์ในโหมดอังกฤษ แต่คีย์นี้คือ จ ในโหมดภาษาไทยจึงใส่ค่า $A6 ผมคิดว่านำรูปที่ 1 และที่ 2 มาเปรียบเทียบกับตารางนี้ก็จะทำความเข้าใจได้ไม่ยาก (สำหรับผู้ที่ยังไม่คุ้นกับเลขฐานสิบหก บรรทัดแรกคืออินเด็กซ์ $20 ถึง $2F บรรทัดที่ 2 คือ $30 ถึง $3F และต่อๆ ไป จนถึงบรรทัดสุดท้ายคือ $70 ถึง $7F)

เมื่อทำตารางแล้ว ตัวโปรแกรม getc จึงเขียนได้สั้นอย่างที่เห็น ขอทำความเข้าใจก่อนครับว่า การเรียกใช้โปรแกรม getc ส่วนมากจะทำอยู่ในวงรอบ ซึ่ง getc จะอ่านค่าจากคีย์บอร์ดมา 1 ตัว แล้วนำมาตรวจดูว่าเป็นคีย์ TOGGLE หรือไม่ ถ้าใช่จะกลับแฟลก Tflg จากจริงเป็นเท็จหรือกลับกัน (ถ้า Tflg เป็นจริงหมายถึงอยู่ในโหมดภาษาไทย) ส่งสัญญาณเสียง 1 ครั้งเพื่อบอกให้ผู้ใช้ทราบว่ากดถูกต้อง เสร็จแล้วให้ค่าคีย์เป็น Null คือ $00 ทั้งนี้เพื่อคุณอาจใช้คีย์ ~ ซึ่งมีรหัสอักขระเป็น # $7E ได้โดยไม่ยุ่งกับคำสั่งในถ้อยแถลงต่อไป

ถ้อยแถลงที่สองเป็นการตรวจว่า ถ้าเป็นคีย์อักษรที่ไม่ใช่คีย์ควบคุม และถ้าอยู่ในโหมดภาษาไทย จะเปลี่ยนค่าคีย์จากอักขระอังกฤษเป็นไทยโดยเปิดตารางดังที่กล่าวมา และได้เปลี่ยนค่าจากเลขจำนวนเป็นรหัสอักขระด้วย โดยใช้คำสั่ง Chr

ถ้าเราไม่ต้องจัดอักขระในภาษาไทยเป็น 3 ระดับ เรื่องราวของการทำภาษาไทยก็จบเพียงเท่านี้ อีกเรื่องหนึ่งที่ผมขอกล่าวให้ชัดเจนคือคำว่า โหมดไทย–อังกฤษ ใช้ในการอ่านคีย์เท่านั้น ไม่เกี่ยวกับการออกจอภาพ

โปรแกรมย่อยอิสระ

ในการนำภาพอักขระออกจอภาพในแบบ 3 ระดับนั้น เราจะใช้โปรแกรม putc ใน putc มีโปรแกรมย่อยรองลงไปอยู่ 4 ตัว คือ putMid, putUnder, putUpr และ Combine ซึ่ง 3 โปรแกรมแรกมีโปรแกรมอื่นเรียกใช้ด้วย จึงต้องนำออกมาจาก putc ให้เป็นโปรแกรมย่อยอิสระที่โปรแกรมใดจะเรียกใช้ก็ได้ ผมจึงขอกล่าวถึงโปรแกรมย่อยเหล่านี้ก่อน

ในการนำภาพอักษรไปออกจอภาพเป็น 3 ระดับนี้ ผมใช้หลักการจัดตัวแปร x และ y ติดตามตำบลที่ของเคอร์เซอร์ ซึ่งในภาษาปาสคาล(และโดยทั่วไป) เคอร์เซอร์จะอยู่ ณ ตำแหน่งที่เราจะพิมพ์ภาพอักขระ เมื่อพิมพ์แล้วเคอร์เซอร์จะอยู่หลังอักขระนั้น ซึ่งเราต้องเลื่อนตัวแปร x ตาม ดังในโปรแกรม putMid(รูป B) ซึ่งจะพิมพ์ภาพอักขระพวกพยัญชนะ หรืออักขระในภาษาอังกฤษที่บรรทัดกลาง สำหรับคำสั่งที่ให้ UpCntr

B

```
(***** putc MODULE ******)

PROCEDURE putMid(ch : Char);
BEGIN
  Write(ch);
  x := x + 1;
  UPcntr := 0;
END;

PROCEDURE putUnder(ch : Char);
BEGIN
  IF y = 25 THEN
  BEGIN WriteLn;  y := y - 1 END;
  gotoXY(x - 1,y + 1);
  Write(ch);
  gotoXY(x,y);
END;

PROCEDURE putUpr(ch : Char);
BEGIN
  gotoXY(x - 1,y - 1);
  write(ch);
  gotoXY(x,y);
END;
```

เขียนไทย

เป็นศูนย์จะกล่าวต่อไป

ส่วนเมื่อเราจะพิมพ์อักขระใดบนหรือล่างพยัญชนะ ทำโดยเลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ต้องการ พิมพ์ภาพอักขระ แล้วเลื่อนเคอร์เซอร์กลับไปยังตำแหน่งหลังพยัญชนะตามเดิม ขอให้ดูในโปรแกรม PutUpr ก่อน จะเห็นว่าใช้คำสั่ง gotoXY เลื่อนเคอร์เซอร์ถอยหลัง 1 อักขระ และขึ้นบน 1 บรรทัดอักขระ เมื่อพิมพ์แล้วเลื่อนเคอร์เซอร์กลับที่เดิม

ใน putUnder จะเป็นการถอยหลังและลงล่าง แต่เราจำเป็นต้องตรวจดูก่อนว่าจะลงล่างได้หรือไม่ คือเคอร์เซอร์อยู่ที่บรรทัดที่ 25 หรือไม่ ถ้าใช่เราต้องเลื่อนภาพทั้งจอภาพขึ้นบน 1 บรรทัดอักขระ โดยใช้คำสั่ง WriteLn เมื่อใช้คำสั่งนี้แล้วบรรทัดเดิมจะกลายเป็นบรรทัดที่ 24 ไป จึงต้องให้ค่า y เป็น 24 ด้วย และเคอร์เซอร์จะอยู่ที่ต้นบรรทัดที่ 25 แต่ถ้อยแถลงต่อไปจะเป็นการนำเคอร์เซอร์ไปไว้ ณ ตำแหน่งที่จะพิมพ์ภาพอักขระล่าง เมื่อพิมพ์แล้วให้ไปอยู่หลังพยัญชนะตามเดิม

putc

โปรแกรมย่อย putc จะนำอักขระ 1 ตัวออกจอภาพในแบบจัดระดับ 3 ระดับ ดังนั้นจะรวมสระบนกับวรรณยุกต์ หรือไม้ทัณฑฆาตเป็นอักขระผสม การรวมกันนี้จะรวมเฉพาะที่แสดงบนจอภาพเท่านั้น ไม่เปลี่ยนแปลงข้อมูลเดิม เพื่อทำความเข้าใจโปรแกรมนี้ ผมขอกล่าวก่อนว่า เราอาจจะใช้ getc เพื่ออ่านอักขระไทยเพียง 1 ตัวได้อย่างคุ้มค่ากับเวลาที่ใช้ แต่สำหรับ putc แล้วไม่คุ้มค่าเพราะสามารถใช้คำสั่ง Write ได้โดยตรง

การเขียนโปรแกรมนี้ จึงวางแนวความคิดไว้ว่า เพื่อให้เรียกใช้อย่างต่อเนื่องเช่นในวงรอบ(loop) จึงได้กำหนดตัวแปร UpCntr ให้เป็นตัวนับ ซึ่งโดยปกติเราจะพิมพ์อักขระบนพวกนี้หลังพยัญชนะแล้ว ดังนั้นจึงให้ putMid ทำให้ UpCntr เป็นศูนย์ (วรรคหรือ space เป็นพยัญชนะตัวหนึ่ง)

โปรแกรม putc อยู่ในรูป C และ D ขอให้ดูในรูป D ก่อน putc จะตรวจว่ารหัสอักขระที่ส่งมาเป็นอักขระที่จะนำออกจอภาพหรือไม่ (คือไม่ใช่พวกคีย์ควบคุม) ถ้าเป็นอักขระที่นำออกจอภาพ จะตรวจว่าเป็นอักขระที่พิมพ์บรรทัดกลางหรือไม่ (คือพยัญชนะ สระบางตัว และอักขระในภาษาอังกฤษ) ถ้าใช่ให้นำออกจอภาพในบรรทัดกลางโดยเรียกใช้ putMid ถ้าไม่ใช่ให้ตรวจว่าเป็นอักขระล่าง(อุกับอู) หรือไม่ ถ้าใช่ให้นำออกจอภาพบรรทัดล่างด้วยการเรียกใช้ putUnder

ถ้าไม่ใช่อีก จะเป็นสระบนและวรรณยุกต์ (รวมทั้งไม้ทัณฑฆาต) จะให้พิมพ์บรรทัดบน โดยถ้าเป็นตัวแรกให้นำออกจอภาพบรรทัดบนโดยเรียกใช้ put Upr ถ้าเป็นตัวที่สองของพยัญชนะตัวนั้น จะนำไปทำเป็นอักขระผสมก่อนโดยการ

C

```
PROCEDURE  putc( ch : Char);

  PROCEDURE  Combine(old : Char; VAR new : Char);
  CONST  UPtab : ARRAY[#$D9..#$DE] OF Byte =
           ($EE,$F3,$F7,$FB,$EA,$E6);
         sara : SET OF Char = [#$D9..#$DF];
         vanayuk : SET OF Char = [#$E0..#$E4];
  VAR    old1,new1 : Byte;
  BEGIN
  IF (old IN sara) AND (new IN vanayuk) THEN
    BEGIN
    old1 := UPtab[old];
    new1 := ord(new) - $E0;
    new  := Chr(old1 + new1);
    END
  ELSE
    BEGIN
    GoodCh := False; new := #$00;
    UPcntr := 0; Write(BEEP)
    END;
END;
```

D

```
BEGIN  { PUTC }
 IF ch >= SPCE THEN
  IF ch < UNDR THEN
    putMid(ch)
  ELSE
    IF ch < UPR THEN
      putUnder(ch)
    ELSE
      BEGIN
      UPcntr := UPcntr + 1;
      IF UPcntr = 1 THEN
        BEGIN putUpr(ch); OldCh := ch END
      ELSE
        IF UPcntr = 2 THEN
          BEGIN Combine(OldCh,ch); putUpr(ch) END
        ELSE
          BEGIN GoodCh:= False;  Write(BEEP) END;
      END;
END;
```

เขียนไทย

เรียกใช้ Combine แล้วจึงนำมาออกจอภาพ โดยเรียกใช้ putUpr เช่นผู้ใช้ต้องการพิมพ์คำว่า นี่ ก็จะคีย์ น สระอี และไม้เอกตามลำดับ สระอีจะเป็นอักขระบนตัวแรกของ น และไม้เอกเป็นอักขระบนตัวที่สองของ น แต่ถ้าผู้ใช้กดอักขระบนต่อไปอีกเช่นคีย์ไม้โท จะไม่ดำเนินการใด แต่จะทำให้แฟลก GoodCh เป็นเท็จ และส่งเสียงเตือน แฟลก GoodCh ผมจะนำไปใช้ต่อไป

Combine

ในการผสมสระกับวรรณยุกต์เข้าด้วยกัน เราจะต้องวางหลักเกณฑ์ให้ผู้ใช้ทราบว่า จะต้องพิมพ์สระก่อนวรรณยุกต์ หรือจะยอมให้พิมพ์อะไรก่อนก็ได้ โปรแกรมนี้กำหนดให้พิมพ์สระก่อน ก็เพื่อให้เขียนโปรแกรมได้สั้นและอ่านง่ายเท่านั้นครับ

สำหรับการผสมสระกับวรรณยุกต์ทางห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์ ม.เกษตรศาสตร์ ผู้กำหนดตารางรหัสอักขระภาษาไทย ดังที่ผมนำมาแสดงในรูปที่ 2 ได้วางรูปแบบไว้แล้ว ผมเพียงแต่นำมาเขียนเป็นโปรแกรม ขอให้ดูในรูปนี้จะเห็นว่าวรรณยุกต์เอกถึงจัตวาและไม้ทัณฑฆาต จะมีรหัสอักขระเป็น $E0 เป็น $E4 ส่วนสระผสมจะแบ่งกลุ่มไว้ตามรูปของสระและเรียงตามวรรณยุกต์ หากผมนำจุดเริ่มต้นมาบวกกับรหัสของวรรณยุกต์ที่ลบด้วย $E0 แล้ว ก็จะได้รหัสของอักขระผสมนั้น เช่นไม้ผัดกับไม้โท จะได้เป็น $EA + ($E1-$E0) ได้ $EB

แต่จุดเริ่มต้นนี้กระจายกันอยู่และไม่เรียงตามรูปสระ จึงต้องทำเป็นตารางให้ชื่อว่า UPtab โดยใช้ลำดับของสระเป็นอินเด็กซ์ ในโปรแกรมจึงเพียงแต่ตรวจว่าอักขระแรกเป็นสระหรือไม่ และอักขระตัวที่สองเป็นวรรณยุกต์หรือไม่ ถ้าเป็นจริงทั้งคู่ก็จะผสมให้ตามกรรมวิธีที่กล่าวมา ถ้าไม่ใช่จะให้แฟลก GoodCh เป็นเท็จ ให้ new เป็น null ซึ่งจะผ่านค่ากลับไปให้แก่ ch ใน putc ให้ UpCntr เป็นศูนย์ และส่งเสียงเตือน 1 ครั้ง

จะเห็นว่า Combine จะผ่านอักขระผสมกลับไปให้ putc ใน ch ซึ่ง putc จะนำออกพิมพ์ในบรรทัดบนทุกครั้ง ไม่ว่าจะเป็นรหัสของอักขระผสมที่ถูกต้องหรือเป็น null ซึ่งถ้าเป็น null ก็จะเห็นเป็นว่าลบภาพอักขระเดิมบนจอภาพ เช่นเดิมมีไม้เอกอยู่ก็จะถูกลบไป จึงต้องให้ Up Cntr เป็นศูนย์ วิธีเขียนโปรแกรมแบบนี้เขียนง่าย แต่รูปแบบของการเขียนโปรแกรมไม่ดีนัก เพราะ putc ควรจะมีหน้าที่เพียงแต่นำภาพอักขระออกจอภาพ ไม่ใช่ไปลบของเดิมเขา

ReadStr และ WriteStr

ในการใช้งานจริง getc กับ putc มีที่ใช้น้อย งานส่วนมากมักจะเป็นการอ่าน-เขียนข้อมูลเป็นสตริง ซึ่งในกรณีของภาษาไทยเราจะใช้สั่ง Readและ Write ไม่ได้ ต้องสร้างโปรแกรมย่อยขึ้นมาดำเนินการในเรื่องนี้เอง คือ ReadStr และ WriteStr

สำหรับ ReadStr อยู่ในรูป E และ F จะอ่านคีย์โดยเรียกใช้ getc และนำออกแสดงทางจอภาพโดยใช้ putc จนกว่าผู้ใช้จะกดคีย์ ENTER ค่าที่อ่านมาจะเก็บในตัวแปรสตริง stn ซึ่งจะผ่านค่ากลับไปให้แก่โปรแกรมที่เรียกใช้ แต่ในการทำงานเกี่ยวกับสตริง ผมใช้กรรมวิธีของ ARRAY ตามหลักการของภาษาปาสคาลที่ว่า String[50] คือ

ARRAY [0....50] OF Char

คือสตริงนี้จะมีช่องเก็บอักขระอยู่ 51 ช่อง โดยที่ช่องแรกเป็นช่องสำหรับเก็บจำนวนอักขระที่มีจริงในสตริงนั้น

ขอให้ดูในรูป F ขึ้นต้นด้วยการเตรียมค่าต่างๆ จะเห็นว่าให้แฟล็ก Tflg เป็นจริง ดังนั้นคีย์แรกที่ผู้ใช้ป้อนจะเป็นอักษรไทย (หากต้องการให้เป็นอักษรอังกฤษผู้ใช้ต้องกดคีย์ CTRL-T เปลี่ยนไปมา) เสร็จแล้วเข้าวงรอบ UNTIL เรียกใช้ getc กับ putc ทำให้ผู้ใช้สามารถป้อนคีย์ได้ และนำออกแสดงบนจอภาพ และดังที่ทราบแล้วว่า ถ้า

E

```
(*******  ReadString  MODULE  *******)
PROCEDURE  ReadStr(VAR stn : anyStr);
VAR  idx : Byte;  ch : Char;  OK : Boolean;

  PROCEDURE  ClrOne;
  VAR  ch : Char;
  BEGIN
    IF idx > 1 THEN
      BEGIN
      idx := idx - 1;  ch := stn[idx];
      IF ch < UNDR THEN    { mid row }
        BEGIN
        x := x - 1;  gotoXY(x,y);
        Write(SPCE);  gotoXY(x,y);
        END
      ELSE
        IF ch < UPR THEN
          putUnder(SPCE)
        ELSE
          BEGIN
          putUpr(SPCE);
          ch := stn[idx - 1];
          IF ch >= UPR THEN
            BEGIN putUpr(ch); UPcntr := 1; OldCh := ch END
          ELSE
            UPcntr := 0;
          END;
      END;
  END;
```

```
BEGIN   { ReadStr }
  UPcntr := 2;  Tflg := True;  idx := 1;
  x := WhereX;  y := WhereY;
  stn[0] := SPCE;
  REPEAT
    GoodCh := True;
    getc(ch);  putc(ch);
    IF NOT GoodCh THEN ClrOne;
    IF (ch >= SPCE) AND GoodCh THEN
      BEGIN
       stn[idx] := ch;
       idx := idx + 1;
      END
    ELSE
      CASE ch OF
       BS : ClrOne;
       RET : stn[0] := chr(idx - 1);
      END;
  UNTIL (ch = RET);
END;
```

F

```
(********  WriteStr  MODULE  *********)
PROCEDURE  WriteStr(stn : anyStr);
VAR  idx,max: Byte;  ch : Char;
BEGIN
  max := ord(stn[0]);
  x := WhereX;  y := WhereY;
  FOR idx := 1 TO max DO
    BEGIN
    ch := stn[idx];
    putc(ch);
    END;
END;
```

G

ผู้ใช้ป้อนคีย์ผิดจะได้ GoodCh เป็นเท็จและ getc จะลบภาพอักขระบนจอภาพ ReadStr ก็ต้องยกเลิกค่าใน stn ด้วย โดยเรียกใช้ ClrOne

จากนั้นก็นำรหัสอักขระมาตรวจดู ว่าเป็นพวกคีย์ที่ออกจอภาพหรือไม่ ถ้าใช่ก็นำไปเก็บไว้ในตัวแปรสตริง stn โดยใช้กรรมวิธีแบบอินเด็กซ์ หากไม่ใช่ก็เป็นคีย์ควบคุม ซึ่งผมสนใจอยู่ 2 คีย์เท่านั้น คือ RET(ENTER) กับ BS(back space) ถ้าเป็นคีย์ BS แสดงว่าผู้ใช้ต้องการลบอักขระที่ได้ป้อนเข้ามาแล้ว ทำโดยเรียกใช้ ClrOne ดังที่จะกล่าวต่อไป ถ้าเป็นคีย์ RET แสดงว่าผู้ใช้ต้องการจบการป้อน ผมนำค่าอินเด็กซ์ idx มาทำเป็นจำนวนอักขระที่ผู้ใช้ป้อนเข้ามาทั้งหมดไปเก็บไว้ต้นสตริง แต่การเก็บต้องแปลงเป็นรหัสอักขระ เพราะสตริงคืออะเรย์ของอักขระดังกล่าวแล้ว

ตรงนี้ผมใช้ CASE เผื่อว่าถ้าคุณผู้อ่านต้องการจะเพิ่มคีย์ควบคุมใดๆ อีก ซึ่งถ้าเพิ่มมากๆ ก็จะกลายเป็นโปรแกรมประเภทเวิร์ด-สตาร์ไป สำหรับโปรแกรม Write Str ในรูป G ก็เพียงแต่อ่านค่าสตริง stn ในแบบอะเรย์ และนำออกแสดงทีละอักขระโดยการเรียกใช้ putc หากคุณจะต่อเติมโปรแกรม ReadStr ควรนำ WriteStr ไปไว้ก่อน ReadStr เพราะอาจจำเป็นต้องใช้ Write Str ใน Read Str

ClrOne

ความมุ่งหมายหลักของการเขียนโปร-แกรม ClrOne ก็เพื่อใช้เมื่อผู้ใช้ต้องการลบอักขระที่ตนพิมพ์ไปแล้ว แต่ผมได้นำไปใช้เมื่อผู้ใช้ป้อนคีย์ผิดด้วย จึงมีบางตอนที่เป็นการลบจอภาพบนจอภาพซ้อนกัน 2 ครั้ง

ผมขอทบทวนว่า ถึงแม้เราจะแสดงภาพอักขระผสมบนจอภาพ แต่การเก็บข้อมูลในสตริงนั้นเราเก็บแยกกัน เช่น คำว่า "นี้" จะเก็บเป็น #$B7, #$DA, #$E1 ไม่ใช่เก็บเป็น #$B7, #$F4 สำหรับแบบการลบข้อมูลที่เขียนโปรแกรมได้ง่ายคือ ให้ลบทีละอักขระต่อการกดคีย์ BS หนึ่งครั้ง เช่นคำว่า "นี้" ตามที่ยกมา จะลบไม้โทก่อนเหลือเป็น "นี" แล้วลบสระอีเหลือเป็น "น" แล้วจึงลบตัว น

การลบต้องลบทั้งภาพบนจอภาพและข้อมูลในสตริง การลบข้อมูลในสตริงทำโดยเลื่อนอินเด็กซ์ถอยหลังมา 1 อักขระ คือ idx := idx-1 ก็เป็นการลบข้อมูลในสตริงโดยสมบูรณ์แล้ว

ส่วนการลบข้อมูลบนจอภาพ ทำโดยเลื่อนเคอร์เซอร์ไปตรงอักขระที่จะลบ แล้วพิมพ์ space ซึ่งทำได้กับอักขระในบรรทัดกลางและบรรทัดล่าง แต่ถ้าเป็นอักขระบนอาจเป็นอักขระผสม เช่น สระอีกับไม้โท การลบครั้งแรกเป็นการลบไม้โท แต่ที่เราลบบนจอภาพไปนั้นเป็นการลบทั้งหมด จึงต้องพิมพ์ภาพสระอีออกจอภาพอีกครั้ง รายละเอียดขอให้ดูในโปรแกรมในรูป E

แสดงคำเตือน

เป็นอันว่าคุณได้โปรแกรมย่อยที่จะทำให้ผู้ใช้โปรแกรมของคุณ สามารถป้อนข้อมูลเป็นภาษาไทย เพื่อนำมาดำเนินกรรมวิธี และสามารถนำออกแสดงบนจอภาพได้แล้ว การป้อนโปรแกรมนี้ขอให้ป้อนเรียงลำดับจากรูป A ถึง G โดยให้ชื่อไฟล์ว่า THULT.SBR

แต่ผมยังไม่ได้กล่าวถึงการแสดงคำเตือน ที่ภาษาอังกฤษเรียกว่า prompt เช่น "enter data" ซึ่งในภาษาไทยคงจะใช้คำว่า "โปรดป้อนข้อมูล" หรือข้อความใดก็ตามที่ต้องการให้แสดงบนจอภาพ ซึ่งผมขอรวม

เขียนไทย

เรียกว่า **คำเตือน** การแสดงคำเตือนทำได้ 3 วิธี ดังแสดงในรูปที่ 4 คือ

1. ให้ค่าคงตัวเป็นรหัสอักขระ แล้วใช้ WriteStr พิมพ์ออกจอภาพ คือ PRMPT1 และ PRMPT1-1 ในรูป วิธีนี้ค่อนข้างยุ่งยากแต่อาจจำเป็นต้องใช้ ถ้าไม่มีโปรแกรมดังที่จะกล่าวในข้อต่อไป

2. ใช้โปรแกรม THAI.COM บางแผ่นอาจชื่อ THAIKU.COM ของ ม.เกษตรศาสตร์ ซึ่งต้องรันโปรแกรมนี้ก่อนรันเทอร์โบปาสคาล เมื่อรันจะแสดงคีย์ที่จะใช้ให้จดไว้ เฉพาะ F1 ถึง F4 ก็เพียงพอ ซึ่งจะเป็นการเลือกใช้โหมดไทย/อังกฤษ แสดง 1 หรือ 3 ระดับ โปรแกรมนี้จะฝังตัวอยู่ในหน่วยความจำ เมื่อรันเทอร์โบปาสคาลแล้ว สามารถเลือกให้เป็นโหมดไทยหรืออังกฤษได้ด้วยการกดคีย์ Alt กับ F ที่กล่าวมา การป้อนโปรแกรมคงดำเนินการตามปกติ สำหรับการแสดงคำเตือนเป็นภาษาไทยทำได้ 3 แบบคือ

– ให้เป็นค่าคงตัว เช่น PRMPT2 และใช้ WriteStr ในการพิมพ์ออกจอภาพ ที่ในโปรแกรมให้พิมพ์เลข 2 นำหน้า

– ให้เป็นค่าคงตัวในคำสั่ง WriteStr โดยป้อนข้อความในแบบ 1 บรรทัด ที่ในโปรแกรมให้พิมพ์เลข 3 นำหน้า

– ให้เป็นค่าคงตัวในคำสั่ง WriteLn ขอให้ดูในโปรแกรมที่ให้พิมพ์เลข 4 นำหน้า แบบนี้ทำให้อ่านโปรแกรมง่าย แต่ยุ่งยากในการป้อนโปรแกรมพอสมควร โดยเฉพาะเมื่อต้องพิมพ์ค่าอื่นอีกในบรรทัดเดียวกัน

3. ใช้โปรแกรมประเภทเวิร์ดสตาร์ภาษาไทย เช่น ขวัญใจนักพิมพ์ดีด สร้างซอร์สโปรแกรมขึ้นก่อน ส่วนการป้อนคำเตือนคงป้อนตามที่กล่าวในข้อ 2

ทั้ง 3 วิธีนี้ วิธีที่ 2 เป็นวิธีที่สะดวกที่สุด ผมคิดว่าคุณคงหาโปรแกรม THAI.COM ได้ แต่ถ้าหาไม่ได้ และเพื่อให้โปรแกรมอรรถประโยชน์ในภาษาปาสคาลของผมสมบูรณ์ยิ่งขึ้น ผมจะนำเวิร์ดสตาร์ภาษาไทยฉบับจิ๋วคือ "คำดาวน้อย" มาเสนอ

โปรแกรมในชุด THULT.SBR นี้เป็นโปรแกรมย่อยที่สามารถนำไปใช้งานได้จริง แต่ผมก็เขียนโปรแกรมนี้เพื่อลงวารสาร จึงต้องการให้สั้นและอ่านง่าย มีข้อบกพร่องที่ควรปรับปรุงเรื่องสำคัญคือ ไม่ควรกำหนดค่าต่างๆ ที่ปรากฏในรูป A เป็นส่วนกลาง ซึ่งจะกลายเป็นส่วนกลางของโปรแกรมของคุณด้วย ควรใช้กรรมวิธีของการผ่านค่าเป็นหลัก และเรื่องที่ putc ทำงานเกินหน้าที่ ขอให้ลองปรับปรุงครับ แต่ถึงอย่างไรโปรแกรมนี้ก็ใช้งานได้เป็นอย่างดี

```
  type demprom.pas
PROGRAM DemoPrompt(output);
TYPE    anyStr = String[50];

{$I thult.sbr}


CONST   PRMPT1 : String[20] = ##D2##B9##C1##B2##B9##E1##CA;
      PRMPT1_1 : String[20] = ##B7##A2##E1##CA##BF##D8##C3;
        PRMPT2 : String[20] = 'โปรดป้อนข้อมูล';
var       astr : anyStr;
PROCEDURE nl;
BEGIN
  WriteLn; WriteLn;
END;

begin  { MAIN }
  WriteLn;
  Write(1); WriteStr(PRMPT1);  WriteStr(PRMPT1_1); nl;

  Write(2); WriteStr(PRMPT2); nl;

  Write(3); WriteStr('โปรดป้อนข้อมูล'); WriteLn;

  Write('  ');WriteLn('     ้  ้');
  Write(4); WriteLn('โปรดป้อนข้อมูล');
  Write('  ');WriteLn('        ู');
end.


Running
     ้  ้
1โปรดป้อนข้อมูล
     ้  ้ ู
2โปรดป้อนข้อมูล
     ้  ้ ู
3โปรดป้อนข้อมูล
     ้  ้ ู
4โปรดป้อนข้อมูล
        ู

>
```

รูปที่ 4 โปรแกรม DemoPrompt แสดงการป้อนคำเตือน และผลการรัน